

(Concise explanations of the disclosure)

Japanese laid-open patent publication No. 9-114734

Published on December 1, 1998

A semiconductor device may include a data cache or a data cache system and a store queue serving as a write buffer or a store buffer for data-write instruction or data store instruction. Data write operation to a main memory and data cache operation to a data memory may be made, wherein a store instruction including a write address and data is once held by the store queue for improvement in throughput of the processor.

Store buffer apparatus with two store buffers to increase throughput of a store operation

Patent Number: ☐ US5845321

Publication date: 1998-12-01

Inventor(s): UEHARA KATSUTOSHI (JP); ISOBE TOSHIKO (JP); KAMADA EIKI (JP); ITO MOTOHISA (JP); YAMAMOTO KEI (JP)

Applicant(s):: HITACHI LTD (JP); HITACHI INFORMATION TECHNOLOGY (JP)

Requested Patent: ☐ JP9114734

Application Number: US19960729837 19961015

Priority Number (s): JP19950266947 19951016

IPC Classification: G06F12/00 ; G06F13/00

EC Classification: G06F12/08B6P

Equivalents:

Abstract

A store buffer apparatus connected to a CPU and a main storage unit includes a first buffer for holding a pair of store address and store data in the main storage unit supplied from an operation execution unit of the CPU, a first latch connected to the first buffer means for holding the store address, a second latch connected to the first latch for holding an output of the first latch, a judgment device for comparing an output read out from the address array with an output of the second latch to thereby judge whether the cache hit check for the store address is successful or not and a second buffer for holding the pair of store data and store address having successful cache hit check judged by the judgment device. Occurrence of the state that the store buffer is full is reduced. Two data stored in the second buffer can possess a format into which the two data can be merged.

Data supplied from the **esp@cenet** database - I2

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平9-114734

(43) 公開日 平成9年(1997)5月2日

(51) Int.Cl.⁹

G 0 6 F 12/08

識別記号

3 1 0

片内整理番号

7623-5B

7623-5B

F I

G 0 6 F 12/08

技術表示箇所

C

3 1 0 Z

審査請求 未請求 請求項の数 3 O L (全 21 頁)

(21) 出願番号 特願平7-266947

(22) 出願日 平成7年(1995)10月16日

(71) 出願人 000005108

株式会社日立製作所

東京都千代田区神田駿河台四丁目6番地

(71) 出願人 000233011

日立コンピュータエンジニアリング株式会
社

神奈川県秦野市堀山下1番地

(72) 発明者 伊藤 元久

神奈川県秦野市堀山下1番地 株式会社日
立製作所汎用コンピュータ事業部内

(74) 代理人 弁理士 鈴木 誠

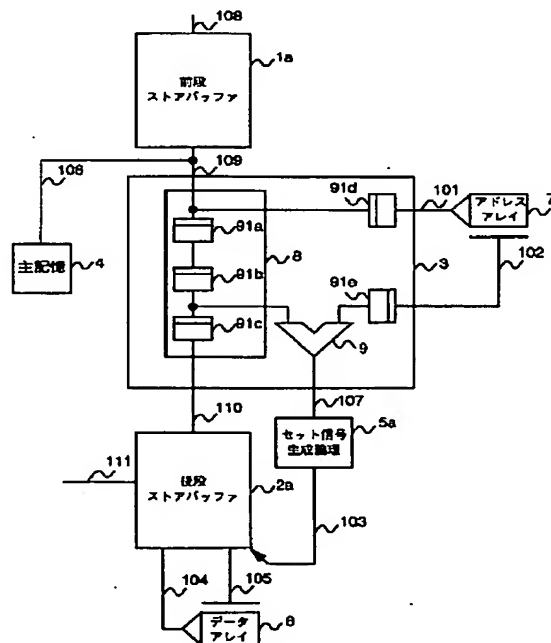
最終頁に続く

(54) 【発明の名称】 ストアバッファ装置

(57) 【要約】

【課題】 キャッシュのヒット判定とデータ書き込みを分離することでキャッシュのヒット判定をパイプライン的に実行し、キャッシュのヒット判定の実行ピッチとストアバッファへの入力ピッチを一致させ、また、キャッシュヒットするストア命令のみストアバッファに入力することにより、ストアバッファに空きがなくなる状態の発生を低減する。

【解決手段】 ストアバッファを前段ストアバッファ1aと、後段ストアバッファ2aに分離し、前段ストアバッファ1aと後段ストアバッファ2aの間でキャッシュのヒット判定3をパイプライン的に行う。キャッシュヒットするストア命令のみを後段ストアバッファ2aに入力する。データアレイ6への書き込みは後段ストアバッファ2aから行う。



【特許請求の範囲】

【請求項1】 演算実行部から出力され、主記憶装置およびキャッシュメモリに書き込むストアアドレスとストアデータとの対を一時保持する複数エントリからなる第1のバッファ手段と、該第1のバッファ手段から取り出したストアアドレスとストアデータとの対が入力され、前記キャッシュメモリとのヒット判定をパイプライン的に行うヒット判定手段と、該ヒット判定の結果、キャッシュヒットと判定されたストアアドレスとストアデータとの対を一時保持する複数エントリからなる第2のバッファ手段と、を備えたことを特徴とするストアバッファ装置。

【請求項2】 前記第1、第2のバッファ手段の1エントリ全てを使用しないデータ長のストアデータを保持するために、該第1、第2のバッファ手段のエントリ内で有効なデータ位置を示す情報を記憶した手段と、前記第2のバッファ手段に入力される第1のストアデータが、該第2のバッファ手段に既に存在する第2のストアデータと同一エントリに併合可能であるか否かを判定する手段と、該手段により併合可能と判定されたとき、該第1のストアデータを該第2のストアデータと併合して該同一エントリに書き込む手段と、該書き込みに応じて、前記記憶された、第2のバッファ手段のエントリ内で有効なデータ位置を示す情報を更新する手段と、を備えたことを特徴とする請求項1記載のストアバッファ装置。

【請求項3】 前記第2のバッファ手段に入力される第1のストアデータが、該第2のバッファ手段に既に存在する第2のストアデータと連結可能であるか否かを判定する手段と、該手段により連結可能と判定されたとき、連結する前記各ストアデータのエントリ番号を記憶する手段と、該記憶されたエントリ番号を基に、前記第2のバッファ手段内の複数エントリのストアデータを連結して前記キャッシュメモリに掃き出す手段と、を備えたことを特徴とする請求項1記載のストアバッファ装置。

【発明の詳細な説明】**【0001】**

【発明の属する技術分野】本発明は、キャッシュメモリを持ち、演算処理装置から主記憶装置及びキャッシュメモリに書き込まれるストアアドレスとストアデータを記憶するストアバッファ装置に関する。

【0002】

【従来の技術】ストア命令が発行されると、キャッシュメモリのヒット判定を行い、キャッシュヒットしたデータがキャッシュメモリに書き込まれる。この従来の方式では、キャッシュのヒット判定の時間が必要なため、書き込み動作のピッチが長くなり、その結果、ストア命令が連続すると中央演算処理装置(以下、CPUと呼ぶ)の性能が低下する。

【0003】この問題点を解決する方法として、データをストアバッファに一旦保持して、キャッシュメモリへ

の書き込み動作はストアバッファへの書き込みをもって終了し、これによりCPUは、データが実際にキャッシュメモリに書き込まれるのを待たずに次の命令の処理に移ることができるキャッシュメモリ制御方法がある(特開平4-37935号公報を参照)。

【0004】上記した従来手法では、最後に書き込んだストアアドレスとキャッシュヒットの判定結果を記憶しておく。そして、後続のストアアドレスと記憶しておいたストアアドレスとを比較し、一致する場合には、キャッシュのヒット判定を行わずに、記憶しておいたキャッシュヒット判定結果を使用する。このようにして、キャッシュのヒット判定を省略することで、書き込み動作のピッチを短くして性能の向上を図っている。

【0005】

【発明が解決しようとする課題】上記公報に記載された方法では、非連続なアドレスにストアする場合、後続のストアアドレスと記憶しておいたストアアドレスが一致せず、記憶しているキャッシュヒット判定を使用することができない。この場合には、改めてキャッシュのヒット判定をしなければならず、書き込みの動作ピッチは向上しない。そのため、ストアバッファは常に空きがない状態になり、バッファとして機能しなくなる。このような状態になると、ストアバッファに空きができるまで、CPUはその実行を待ち合わせるため、CPUの性能が低下するという問題がある。

【0006】また、上記した従来の方法では、ストアバッファから出力される段階で、キャッシュのヒット判定が行われるので、キャッシュミスになるストア命令もストアバッファに保持されていて、ストアバッファのエントリが有効に使用されていない。

【0007】さらに、非連続なアドレスにストアする場合、主記憶への書き込み動作ピッチも長くなり、主記憶につながるバスが有効に使用されないという問題もある。

【0008】本発明の目的は、連続アドレスに対するストアと非連続のアドレスに対するストアの動作ピッチを同じにして、ストアバッファの入力と出力のピッチを一致させ、ストアバッファに空きがなくなる状態の発生を低減させることで、CPUの性能を低下させないようにしたストアバッファ装置を提供することにある。

【0009】本発明の他の目的は、ストアバッファに入力する前にキャッシュのヒット判定を行い、キャッシュヒットするストア命令のみストアバッファに入力することで、ストアバッファの使用効率を向上させたストアバッファ装置を提供することにある。

【0010】本発明のさらに他の目的は、複数のストアバッファエントリのデータを同時にキャッシュに書き込むことで、出力スループットを向上させたストアバッファ装置を提供することにある。

【0011】本発明のさらに他の目的は、キャッシュの

ヒット判定をパイプライン的におこなうことで、主記憶への書き込みピッチとキャッシュのヒット判定のピッチを一致させ、主記憶につながるバスを有効に使用するストアバッファ装置を提供することにある。

【0012】

【課題を解決するための手段】前記目的を達成するために、請求項1記載の発明では、演算実行部から出力され、主記憶装置およびキャッシュメモリに書き込むストアアドレスとストアデータとの対を一時保持する複数エントリからなる第1のバッファ手段と、該第1のバッファ手段から取り出したストアアドレスとストアデータとの対が入力され、前記キャッシュメモリとのヒット判定をパイプライン的に行うヒット判定手段と、該ヒット判定の結果、キャッシュヒットと判定されたストアアドレスとストアデータとの対を一時保持する複数エントリからなる第2のバッファ手段とを備えたことを特徴としている。

【0013】請求項2記載の発明では、前記第1、第2のバッファ手段の1エントリ全てを使用しないデータ長のストアデータを保持するために、該第1、第2のバッファ手段のエントリ内で有効なデータ位置を示す情報を記憶した手段と、前記第2のバッファ手段に入力される第1のストアデータが、該第2のバッファ手段に既に存在する第2のストアデータと同一エントリに併合可能であるかを判定する手段と、該手段により併合可能と判定されたとき、該第1のストアデータを該第2のストアデータと併合して該同一エントリに書き込む手段と、該書き込みに応じて、前記記憶された、第2のバッファ手段のエントリ内で有効なデータ位置を示す情報を更新する手段とを備えたことを特徴としている。

【0014】請求項3記載の発明では、前記第2のバッファ手段に入力される第1のストアデータが、該第2のバッファ手段に既に存在する第2のストアデータと連結可能であるかを判定する手段と、該手段により連結可能と判定されたとき、連結する前記各ストアデータのエントリ番号を記憶する手段と、該記憶されたエントリ番号を基に、前記第2のバッファ手段内の複数エントリのストアデータを連結して前記キャッシュメモリに掃き出す手段とを備えたことを特徴としている。

【0015】上記したように、キャッシュのヒット判定とキャッシュへのデータ書き込みを分離したことにより、後段ストアバッファへの入力と、後段ストアバッファからの出力のピッチが一致し、後段ストアバッファに空きがなくなる状態の発生を低減することができる。また、キャッシュのヒット判定を後段ストアバッファ入力時に行い、キャッシュヒットするストア命令のみ後段ストアバッファに入力することで、後段ストアバッファの使用効率を向上することができる。また、後段ストアバッファの複数のエントリを同時に掃き出せるようにしたことで、後段ストアバッファに空きがなくなる状態の発

生を低減することができる。さらに、キャッシュのヒット判定をパイプライン的に行うことで、主記憶にデータを書き込むピッチとキャッシュのヒット判定のピッチが一致し、主記憶につながるバスを有効に使用することができる。

【0016】

【発明の実施の形態】以下、本発明の一実施例を図面を用いて具体的に説明する。以下の説明では、2進数を“ ”で囲って表すことにする。図25は、本発明の実施対象となる情報処理装置の構成を示す。図中、74はCPU、75はストアスルー方式の2次キャッシュメモリ(以下、SCMと呼ぶ)、4は主記憶装置である。71はプログラムを実行する命令実行部、72はCPU内蔵の1次キャッシュメモリ(以下、FCMと呼ぶ)、73はストアバッファである。6はSCMのデータアレイ、7はSCMのアドレスアレイである。108はストアアドレスとデータ、106は主記憶にデータを書き込むバスである。101はSCMアドレスアレイ7を参照するアドレス、102はSCMアドレスアレイ7の内容、104はSCMストアアドレス、105はSCMストアデータである。

【0017】情報処理装置において、メモリアクセスの高速化のためにキャッシュメモリを用いるのが一般的である。CPUチップ内に設けたFCM72は、高速に動作するが容量を大きくするのは難しい。そこで、FCM72の他に、CPUチップ外にSCM75を設けることがある。このSCM75は、速度はFCM72に劣るが、大容量化が可能である。図25は、CPU74の外にSCM75を設けた場合の構成例である。SCM75は、データアレイ6とアドレスアレイ7からなる。データアレイ6にデータを書き込み、アドレスアレイ7で内容の管理を行う。

【0018】ストア命令の実行は、FCMヒットならばFCM72にデータを書き込み、同時にストアバッファ73に書き込む。ストアバッファ73にデータが書き込まれた段階でストア命令の実行は終了し、命令実行部71は次の命令の処理に移る。ストアバッファに保持されたストア命令は、命令実行部71の動作とは非同期に、主記憶4とSCM75に書き込む。ただし、SCMミスならば、SCM75には書き込まない。

【0019】〈実施例1〉図1は、本発明の実施例1の構成であり、ストアバッファ、SCM、主記憶の書き込みに関係する部分を示す。図1において、1aは前段ストアバッファ、2aは後段ストアバッファ、3はSCMヒット判定論理、4は主記憶、5aはセット信号生成論理である。本発明では、図25のストアバッファ73を、前段ストアバッファ1aと後段ストアバッファ2aに分割し、前段ストアバッファ1aと後段ストアバッファ2aとの間にSCMヒット判定論理3を設けて構成したものである。

【0020】SCMヒット判定論理3は、パイプライン遅延論理8、アドレス比較器9、SCMアドレスアレイ7の参照アドレスを保持するラッチ91d、読み出したアドレスアレイの内容を保持するラッチ91eからなる。パイプライン遅延論理8は、ストアアドレスとデータをパイプライン的に遅延させるためのラッチ91a〜cで構成されている。

【0021】103はセット信号生成論理5aから出力されるセット信号、107はキャッシュヒット判定結果である。109は前段ストアバッファ1aから出力されるストアアドレスとデータであり、110はパイプライン遅延論理8で遅延されたストアアドレスとデータである。111は、指定されたエントリを後段ストアバッファ2aからデータアレイ6に掃きだすための掃き出し要求である。

【0022】図1に示す実施例1の特徴は、ストアバッファが前後2段に分かれていて、前段ストアバッファ1aと後段ストアバッファ2aの間のSCMヒット判定論理3内でパイプライン的にSCMヒット判定を行い、SCMヒットしたストア命令のみを後段ストアバッファ2aに入力することにある。

【0023】図2は、実施例1の前段ストアバッファ1aの構成を示す。主記憶4に書き込む順序を保障するため、前段ストアバッファ1aは、先入れ先出し型(FIFO)のバッファで構成されている。図2において、前段ストアバッファ1aは、本実施例では8段のエントリからなり、各エントリは、本実施例では32ビットのアドレスフィールド53、32ビットのデータフィールド54aからなる。55はリードライトポインタ、122はライトエントリ、123はリードエントリである。

【0024】ストアアドレスとデータ108は、ライトエントリ122が示すエントリに書き込まれ、リードエントリ123が示すエントリのアドレスとデータをセクタ60aで選択して、109に出力する。

【0025】前段ストアバッファ1aから出力したストアアドレスとデータは、主記憶4に送られ、ストアアドレスにデータが書き込まれ、それと同時にSCMヒット判定論理3に入力される。SCMヒット判定論理3は、パイプライン的にSCMヒット判定を行う。

【0026】なお、本実施例のSCMのアドレスマッピングは、例えばダイレクトマッピング法を用いる。図26は、ダイレクトマッピング法を説明する図である。32ビットのストアアドレスの内、上位の例えば12ビット(A0〜A11、タグ)が比較器の一方に入力される。また、例えばストアアドレスA12〜A27の16ビット(インデックス)がアドレスアレイ7の参照アドレスとなって1MブロックのSCMが参照される。そして、下位のA28〜A31の4ビットがブロック内のアドレスとなる。ここでは、1ブロックが16バイトで構成されている。

【0027】アドレスアレイ7には上位12ビット(タグ)が格納され、参照アドレスで読み出される上位12ビットが比較器の他方の入力となる。ストアアドレス中のインデックスを用いてアドレスアレイ7のエントリを指定し、該エントリのタグを読み出す。ストアアドレス中のタグとアドレスアレイ7のタグが比較され(このとき、タグに付けられている有効ビットもチェックされる)、一致したとき、1ブロックまたは、ストアアドレス中のブロック内アドレスで指定された1バイトが読み出される。また、タグが一致しないときは、主記憶をアクセスする。

【0028】図3は、SCMヒット判定論理3の動作例を示す。ラッチ91a〜cで構成されたパイプライン遅延論理8で、ストアアドレスとデータをパイプライン的に遅延する。また、ラッチ91dは、アドレスアレイ7の参照アドレス(つまり、上記したインデックス)を保持する。SCMの一実施例として、図3に示すように、アドレスアレイ参照アドレスを与えてから、3サイクル後にアドレスアレイ7の内容(つまり、上記したタグ)が読み出されるSCMを考える。このSCMは、2サイクルピッチでアドレスを与えることで連続してデータの読み出しが可能なラッチ付きスタックラム(Latched SRAM)で構成されているものとする。アドレスアレイ7の内容がラッチ91eに読み出されるまで、ストアアドレスはラッチ91a〜bで遅延する。ラッチ91bのストアアドレス(タグ)とラッチ91eのアドレスアレイ7の内容(タグ)をアドレス比較器9で比較し、SCMヒット判定を行う。SCMヒット判定結果107は、

SCMヒット: "1"

SCMミス: "0"

の値になる。SCMヒット判定結果107="1"(SCMヒット)ならば、セット信号生成論理5aで後段ストアバッファ2aのセット信号103を生成する。

【0029】このように、本発明ではSCMヒット判定をパイプライン的に行っているため、アドレスアレイ7からアドレスアレイの内容102が出力されると、直ちに前段ストアバッファ1aから次のストア命令をSCMヒット判定論理3に投入することができ、この結果、従来のものに比べてキャッシュのビジー率が低下する。

【0030】図4は、セット信号103を生成するセット信号生成論理5aの構成を示す図である。51は後段ストアバッファ2aのライトポインタ用ラッチ、52は更新論理である。ラッチ51は0から7の値を取り、更新論理52はラッチ51の値を+1する。なお、ラッチ51の値が7のとき、更新論理52は0を出力する。69はデコーダで、ラッチ51の値により以下の値を出力する。

【0031】

ラッチ51の値	デコーダ69の出力
0	"10000000"
1	"01000000"
2	"00100000"
3	"00010000"
4	"00001000"
5	"00000100"
6	"00000010"
7	"00000001"

103は、後段ストアバッファ2aに入力するセット信号である。セット信号103は、本実施例では後段ストアバッファ2aが8段エントリであるので8ビットで構成され、"1"が立っている後段ストアバッファ2aのエントリにストアアドレスとストアデータ、および有効ビットが入力される。つまり、セット信号103の0ビット目は0番エントリに入力することを示し、以下順に、7ビット目が7番エントリに入力することを示す。例えば、3番エントリに入力するセット信号103は、"00010000"になる。セット信号103がオール"0"のときは、後段ストアバッファ2aのどのエントリにも入力されない。このセット信号103は、以下のように生成される。

【0032】(1) SCMヒット判定結果107="1" (SCMヒット) のとき、セレクトは更新論理52を選択し、従ってセット信号103としてデコーダ69の出力がアンドゲートを介して出力される。また、ラッチ51の値は更新論理52により更新される。

【0033】(2) SCMヒット判定結果107="0" (SCMミス) のとき、セレクトは"0"を選択し、従ってセット信号103としてオール"0"が出力され、ラッチ51の値は更新されない。

【0034】図5は、前段ストアバッファ1a内のリードライトポイント55の構成を示す図である。56aはライトポイント用ラッチ、56bはリードポイント用ラッチ、52は前述した更新論理、69は前述したデコーダである。128は命令実行部からのライト要求信号、129は命令実行部からのリード要求信号である。

【0035】本実施例では前段ストアバッファ1aが8段のエントリからなるので、ライトポイント用ラッチ56a、リードポイント用ラッチ56bともに0から7の値をとる。ライト要求信号128が"1"になるたびに、前段ストアバッファ1aにストアアドレス、データが書き込まれ、ライトポイント用ラッチ56aは次の値を示すように更新される。また、リード要求信号129が"1"になるたびに、前段ストアバッファ1aからストアアドレス、データが掃き出され、リードポイント用ラッチ56bは次の値を示すように更新される。なお、更新論理52は、ライトポイント用ラッチ56a、あるいは、リードポイント用ラッチ56bの値が7のときは0を出力する。

【0036】図6は、実施例1の後段ストアバッファ2aの構成を示す図である。図6において、本実施例では後段ストアバッファ2aは、8段のエントリからなり、各エントリは、本実施例では30ビットのアドレスフィールド57a、32ビットのデータフィールド58a、1ビットの有効ビット59からなる。

【0037】アドレスフィールドが前段ストアバッファ1aと同じ32ビット構成でないのは、本実施例ではデータフィールド58aが32ビット構成(4バイト)であり、かつ、アライン(アドレス調整)されているので、ストアアドレスの下位2ビット(30、31ビット目)はSCM書き込み時に使用しないためである。つまり、データアレイの1ブロックが4バイト構成されている。

【0038】61は後段ストアバッファ2aのリードポインタ、62はリードポインタ用ラッチ、52は前述した更新論理、69は前述したデコーダである。ラッチ62は0から7の値を取り、更新論理52はラッチ62の値を+1する。なお、ラッチ62の値が7のとき、更新論理52は0を出力する。

【0039】103は前述したセット信号生成論理5aからのセット信号、111は命令実行部からの掃き出し要求信号、104はSCMデータアレイの書き込みアドレス、105はSCMデータアレイの書き込みデータである。

【0040】パイプライン遅延論理3で遅延したストアアドレスとデータは、信号線110から入力され、ストアアドレスはアドレスフィールド57a内のセット信号103で示すエントリに入力され、データはデータフィールド58a内のセット信号103で示すエントリに入力される。同時に、当該エントリの有効ビット59を"1"にセットする。セット信号103がオール"0"ならば、どのエントリにもストアアドレスとデータは入力しない。このとき、有効ビット59は変化しない。すなわち、SCMミスになるストア命令は、後段ストアバッファ2aには入力されない。

【0041】一方、後段ストアバッファ2aからSCMへの掃き出しは、掃き出しエントリ306の示すエントリから行われる。掃き出しエントリ306は、本実施例では前段ストアバッファ2aが8段のエントリから構成されているので、8ビットで構成される。掃き出しエントリ306の1ビット目は1番エントリを掃き出すことを示し、以下順に、7ビット目が7番エントリから掃き出すことを示す。例えば、0番エントリを掃き出す掃き出しエントリ306の値は"10000000"になる。掃き出しエントリ306は、以下のように生成される。

【0042】(1) デコーダ69の出力="1" かつ、有効ビット59="1" (有効)、

かつ、
掃き出し要求111="1"（有効）、
ならば、
掃き出しエントリ306の該当ビット="1"
（2）上記以外

掃き出しエントリ306の該当ビット="0"
セクタ60cは、掃き出しエントリ306が示すエン
トリのアドレスフィールド57aからストアアドレスを
選択して信号線104に出力し、データフィールド58
aからデータを選択して信号線105に出力する。掃き
出しエントリ306がオール"0"ならばどのエントリ
も出力しない。ストアアドレスとデータの掃き出しと同
時に、掃き出したエントリの有効ビット59をリセット
する。さらに、ラッチ62の値は更新論理52により更
新する。

【0043】掃き出し要求111は、後段ストアバッ
ファ2aにストア命令があり、かつ、SCMが動作してい
ないときに、SCMコントローラ（図1には記載してい
ない）から出力する。このSCMコントローラは公知技術
で実現可能である。

【0044】図7は、図1の実施例1において、連続し
てストア命令を処理するときのタイムチャートを示す。

Store-1 SCM ヒット：
セット信号103="10000000"：ラッチ55=1
Store-2 SCM ミス：
セット信号103="00000000"：ラッチ55=1
Store-3 SCM ヒット：
セット信号103="01000000"：ラッチ55=2

従って、Store-1のストアアドレスとデータは、
セット信号103が指示する後段ストアバッファ2a内
の、0番目のエントリのアドレスフィールド57a、デ
ータフィールド58aにそれぞれ保持され（また、同時
に有効ビット59もセットされる）、同様に、Stor
e-3のストアアドレスとデータは、後段ストアバッ
ファ2a内の1番目のエントリに保持され、ラッチ55の
値は2になる。

【0048】後段ストアバッファ2aに保持されている
ストア命令は、命令実行部からの掃き出し要求111に
よって、後段ストアバッファ2aからSCMのデータア
レイ6に書き込まれる。すなわち、リードポインタ61
によって指定されたエントリがセクタ60cで選択さ
れる。選択されたエントリのアドレスフィールド57a
がデータアレイ書き込みアドレスとなってデータアレイ
6をアクセスする。ここで、データアレイ書き込みアド
レスは、下位2ビットを除く上位30ビットであり、該
アドレスで指定されたブロック内に、先のリードポイン
タ61で指定されたエントリのデータ（32ビット）が
書き込まれる。なお、図1のSCMは、図26のキャ
ッシュメモリと異なり、アドレスアレイ7とデータアレイ
6に分離しているので、データアレイ6にはアドレスが

本実施例1では、ストア命令は、Store-1からS
tore-3まで3命令を連続して前段ストアバッファ
1aから掃き出す場合を例にしている。そして、Sto
re-1、3はSCMヒットし、Store-2はSC
Mミスしたものとする。また、本実施例では信号線10
6は、2サイクルピッチで主記憶へのストア命令を受け
付け可能であるものとする。

【0045】以下、実施例1の動作を説明すると、主記
憶に書き込むのと同じピッチで、ラッチ91aにストア
アドレスとデータが入力され、ラッチ91dにアドレス
アレイの参照アドレス（ストアアドレス中のインデッ
クス）が入力される。インデックスで指定されたアドレ
スアレイ7の内容（つまり、タグ）は、3サイクル後にラ
ッチ91eに読み出される。すなわち、ラッチ91eの
値は2サイクルピッチで変化する。

【0046】ラッチ91eのアドレスアレイの内容（タ
グ）とラッチ91cの遅延したストアアドレス中のタグ
をアドレス比較器9で比較して、SCMヒット判定結果
107が生成される。その結果、セット信号103、ラ
ッチ55の値は以下のように変化する。

【0047】

割り付けられている。

【0049】〈実施例2〉次に、実施例2について説明
する。通常、整数型のデータは4バイト（32ビット）
幅であることが多いが、浮動小数点データは、倍精度で
あると8バイト（64ビット）の幅が必要である。浮動
小数点データを1エントリに格納できるように、データ
フィールドの幅を8バイトにすると、整数型のデータを
格納する際に空きができて無駄が生じる。

【0050】そこで、データフィールドの幅を8バイト
にし、さらに、同じエントリを使用するストア命令は、
併合して1つのエントリを使用するようにすれば、浮動
小数点データを1エントリに格納できるうえに、整数型
のデータを格納する際の無駄が生じない。同じエントリ
を使用するストア命令を併合して1つのエントリを使用
することを、本実施例ではデータの併合と呼ぶことにす
る。そして、データの併合は、後段ストアバッファに対
して行う。つまり、既に後段ストアバッファに存在する
ストアデータと、後段ストアバッファに入力されるストア
データが同一エントリに併合可能であるとき、データ
の併合を行う。

【0051】データが併合可能か否かは、データフィー
ルドが8バイトであるから、下位3ビットを除く、スト

アドレスの上位29ビット(0~28ビット)を比較することによって同一のエントリであるか否かが判明する。ストアアドレスの比較の結果、データ併合が可能ならば、既存のデータフィールドに新しいストアデータを書き込む。ただし、主記憶4に書き込む順番を保障する目的を持つ前段ストアバッファ1bは、データフィールドの幅を8バイトにしてもデータの併合を行わない。

【0052】データフィールドの幅を8バイトに拡張したので、データフィールド内のどこに有効なデータが存在するかを表す情報を記憶しておく必要が生じる。そこで、有効なデータが存在するバイト位置を表す情報をデータフィールドの各エントリ毎に設ける。これをバイトマスクと呼ぶ。

【0053】バイトマスクの値は、

“1”：有効なデータが存在する

“0”：有効なデータが存在しない

である。例えば、0バイト目から3バイト目まで有効なデータが存在する場合のバイトマスクは、“11110000”となる。

【0054】図8は、本発明の実施例2の構成を示す。実施例1の構成に、前段バイトマスクバッファ11、後段バイトマスクバッファ12、バイトマスク遅延論理13、バイトマスク更新論理14、アドレス一致検出論理15aを付加して構成されている。

【0055】図9は、実施例2の前段ストアバッファ1bの構成を示す図である。主記憶4に書き込む順序を保障するため、前段ストアバッファ1bは先入れ先出し型(FIFO)のバッファで構成され、本実施例では8段のエントリを持つ。そして、データフィールド54bは、本実施例では8バイト(64ビット)幅である。信号線122と信号線123は、前段バイトマスクバッファ11の書き込み、あるいは掃き出しに使用する。また、信号線128、129は、命令実行部からのライト要求、リード要求である。リードライトポインタ55、セクタ60bは、実施例1で説明したものと同様の機能を持つ。

【0056】図10は、前段バイトマスクバッファ11の構成を示す図である。データフィールド54bは、本実施例では1エントリ8バイト構成であるので、バイトマスクの1エントリは8ビットで構成される。そして、前段ストアバッファ1bと同じ8段のエントリを持つ。信号線112から入力されたバイトマスクは、ライトエントリ122の示すエントリに入力される。掃き出しは、前段ストアバッファ1bの掃き出しと同時に、セクタ60fでリードエントリ123の示すエントリのバイトマスクを選択する。

【0057】前段バイトマスクバッファ11から掃き出したバイトマスク113は、バイトマスク遅延論理13でSCMヒット判定が終わるまで遅延させ、バイトマスク更新論理14を通過した後、後段バイトマスクバッ

ファ12に入力される。

【0058】図11は、バイトマスク更新論理14の構成を示す図である。114はバイトマスク遅延論理13で遅延させたバイトマスク、116-0~7は、後段バイトマスクバッファ12に保持されたバイトマスクで、116-0は0番エントリのバイトマスク、以下順に、116-7が7番エントリのバイトマスクである。115-0~115-7は後段バイトマスク12の入力である。115-0は0番エントリの入力であり、以下順に、115-7が7番エントリの入力である。121はデータ併合の有無を示す信号で、

“0”：併合なし、

“1”：併合あり、

を表す。

【0059】データ併合の有無に応じて、後段バイトマスクバッファ12の入力115-0~7を以下のように生成する。すなわち、

(1) 信号線121=“0”(データ併合なし)

後段バイトマスク12の入力=ストアデータのバイトマスク114

(2) 信号線121=“1”(データ併合あり)

後段バイトマスク12の入力=ストアデータのバイトマスク114 or (論理和) 既存のバイトマスク116-0~7

図12は、後段バイトマスクバッファ12の構成を示す図である。後段バイトマスクバッファ12は、本実施例では、8エントリで構成され、各エントリは8ビットで、データフィールド58b内の有効なデータが存在するバイト位置を示す。115-0~7は各エントリの入力であり、116-0~7は各エントリの出力である。前段バイトマスクバッファ11と違い、入力は共通ではなく各エントリ毎に個別になっている。そして、セット信号103の示すエントリのバイトマスクのみが書き込まれ、その他のエントリのバイトマスクは変化しない。

【0060】図13は、実施例2の後段ストアバッファ2bの構成を示す図である。各エントリは本実施例2では、29ビットのアドレスフィールド57b、64ビット(8バイト)のデータフィールド58b、1ビットの有効ビット59からなる。アドレスフィールドが32ビット構成でないのは、データフィールド58bが64ビット構成であるので、つまり、SCMデータアレイの1ブロックが64ビット(8バイト)であり、ブロックエントリがストアアドレスの上位29ビットで指定され、下位3ビット(29~31ビット目)がSCMデータアレイの書き込み時に使用されないからである。

【0061】60dは、アドレスフィールド57b、データフィールド58bから掃き出すエントリを選択するセクタ、63aは、有効なデータが存在するバイト位置のストアデータを取り出すバイトセクタである。

【0062】103はセット信号、104はデータアレ

書き込みアドレス、105は書き込みデータである。110はパイプライン遅延論理3で遅延したストアアドレスとデータ、114はバイトマスク遅延論理14の出力、111は掃き出し要求である。116-0~7は後段バイトマスクバッファ12の内容である。119-0~9はアドレスフィールド57bの内容である。124-0~7は有効ビット59の値である。

【0063】データフィールド58bは、バイト毎にストアデータをセット可能なように構成され、データフィールド58bのセット信号307-0~7は、セット信号307-0が0バイト目のセット信号であり、以下順に、セット信号307-7が7バイト目のセット信号である。

【0064】パイプライン遅延論理3で遅延されたストアアドレスとデータは、信号線110から入力され、ストアアドレスはセット信号103で示すアドレスフィールド57bのエントリに入力され、データはセット信号307-0~7で示すデータフィールド58bのエントリのバイト位置に入力される。ストアアドレスとデータが入力されると同時に、当該エントリの有効ビット59が“1”にセットされる。セット信号103がオール“0”ならば、どのエントリにもストアアドレスとデータは保持されず破棄される。このとき、有効ビット59は変化しない。すなわち、SCMミスになるストア命令は、後段ストアバッファ2bに入力されない。

【0065】一方、後段ストアバッファ2bからの掃き出しは、掃き出しエントリ306が示すエントリからおこなわれる。セクタ60dは掃き出しエントリ306を基に、アドレスフィールド57bからストアアドレスを選択して信号線104に出力し、データフィールド58bからデータを選択して信号線301に出力する。ストアアドレスとデータの掃き出しと同時に、掃き出したエントリの有効ビット59をリセットする。さらに、ラッチ62の値は更新論理52により更新する。バイトセクタ63aは、有効なストアデータを取り出し信号線105に出力する。

【0066】掃き出し要求信号111は、後段ストアバッファ2bにストア命令があり、かつ、SCMが動作していないときに、前述の公知技術であるSCMコントローラ（図8には示されていない）から出力する。

【0067】図14は、バイトセクタ63aの構成を示す図である。60gは掃き出しエントリ306を基に、後段バイトマスクバッファの内容116-0~7から掃き出すエントリのバイトマスク308を取り出すセクタである。このバイトマスク308を基にセクタ60hは、ストアデータ301から有効なデータを取り出す。

【0068】図15は、データの併合を検査するためのアドレス一致検出論理15aの構成を示す図である。64-0~7は、29ビットのアドレス比較器である。1

17は後段ストアバッファ2bに入力可能なストアデータのストアアドレス、119-0~7は、後段ストアバッファ2bに既に存在するストアデータのアドレスフィールド57bの内容、124-0~7は有効ビット59の内容である。

【0069】29ビットのストアアドレス117が、各アドレス比較器64-0~7の一方に入力され、各アドレス比較器64-0~7の他方にはそれぞれアドレスフィールド57bの内容A0~A7が119-0~7として入力され、ストアアドレス117を持つストアデータが、後段ストアバッファ2bに既に存在するどのストアデータと併合が可能であるか否かを調べる。118-0~7は、アドレスが一致しデータ併合可能なことを示す。つまり、118-0は、ストアアドレス117を持つストアデータが、後段ストアバッファ2bの0番エントリのデータと併合可能であることを示し、以下順に、118-7は7番エントリとデータ併合可能であることを示す。

【0070】ただし、アドレス比較器64-0~7は、enable（有効ビット59）が“1”（有効）であるとき、ストアアドレス117の0~28ビットと、アドレスフィールド57bの内容119-0~7をそれぞれ比較する。そして、アドレスが一致したならば、“1”を出力し、不一致ならば“0”を出力する。enable（有効ビット59）が“0”（無効）であるときは“0”を出力する。

【0071】図16は、実施例2のセット信号生成論理5bの構成を示す図である。121は、図15のデータ併合可能なことを示す信号118-0~7をORゲート201を介した信号であり、“1”のときデータ併合があり、“0”のときデータ併合がないことを示す。

【0072】セット信号生成論理5bは、条件により以下のように動作する。すなわち、

（1）信号線107=“1”（SCMヒット）、かつ、信号線121=“0”（データ併合なし）ならば、アンドゲート202の出力“1”によって、セクタ203は更新論理52側の出力を選択する。ラッチ51の値がデコード69で出力され、アンドゲート204、オアゲート205を介して信号103に出力される。この場合はデータの併合がないので、実施例1と同様に、信号103で指示された後段ストアバッファ2bのエントリに入力される。また、この入力時に、前段バイトマスクバッファ11からのバイトマスク114が後段ストアバッファ2bにアンドゲート401を介して入力されるので、そのバイトマスク114で指示されたバイト位置に入力される。また、ラッチ51の値は更新論理52により更新される。

【0073】

（2）信号線107=“1”（SCMヒット）、かつ、信号線121=“1”（データ併合あり）ならば、

アンドゲート202の出力は“0”となり、セクタ203は“0”を選択し、デコーダ69はオール“0”となる。また、アンドゲート204の出力が“0”となる。従って、信号103としては、エン트리番号を示す信号118-0〜7がオアゲート205を介して出力される。

【0074】エン트리番号を示す信号103は、図13の後段ストアバッファ2bに入力され、これと同時にストアデータのバイトマスク114が入力される。これにより、併合されるストアデータは、信号103で指示されたエン트리中の、バイトマスク114で指示されたバイト位置のデータフィールド58bに入力される。

【0075】なお、併合時に、入力ストアデータが、後段ストアバッファ2bに存在する併合相手のストアデータと同じバイト位置に書き込まれ、つまり上書きされることもあるが、同一エントリに対するストア命令が連続したとき、最後のストア命令が後段ストアバッファ2bに保持され、その後、SCMに最新のデータが掃きだされればよいので、何ら問題ではない。

【0076】

(3) 信号線107=“0”(SCMミス)ならば、アンドゲート202の出力は“0”となり、セクタ203は“0”を選択し、デコーダ69はオール“0”となる。従って、信号線103はオール“0”、ラッチ51の値は更新されない。

【0077】〈実施例3〉次に実施例3について説明する。SCMへの書き込みデータ幅を広くすれば、データ書き込みの効率はよくなる。そこで、本実施例ではSCMへの書き込みデータ幅を2倍の128ビット(16バイト)に広げ、後段ストアバッファの2エントリを連結して掃き出すようにする。連結可能であるか否かは(つまり、2エントリのアドレスが連続しているか否かは)、書き込みデータ幅が16バイトであるから、ストアアドレスの上位28ビット(0〜27ビット目)を比較すれば判明する。

【0078】図17は、本発明の実施例3の構成を示した図である。2cは後段ストアバッファであり、15bはアドレス一致検出論理である。16は連結エントリバッファである。連結エントリバッファ16は、後段ストアバッファ2cの複数のエントリを連結して掃き出すとき、どのエントリと連結するかを示す。

【0079】実施例3の特徴は、2エントリを連結して同時に後段ストアバッファ2cから掃き出す制御方法にある。まず、どのエントリと連結可能であることを示す連結エントリバッファ16を持つ。また、ストアアドレスを比較して連結可能なエントリを判定するアドレス一致検出論理15bを持つ。さらに、任意の2エントリを同時に掃き出し可能な後段ストアバッファ2cを持つ。

【0080】図18は、連結エントリバッファ16の構成を示す図である。連結エントリバッファ16は、後段

ストアバッファ2cと同じ数のエントリを持ち、本実施例では8段のエントリから構成され、各エントリは8ビットの連結エントリフィールドME_nからなる。

【0081】連結エントリフィールドME_nは、n番エントリと連結可能なエントリを示し、0ビット目が0番エントリと連結可能であることを示し、以下順に、7ビット目が7番エントリと連結可能であることを示す。

【0082】例えば、1番エントリと3番エントリが連結可能ならば、連結エントリフィールドME₁には、3番エントリと連結可能であることを示す、ME₁=“00010000”が格納され、連結エントリフィールドME₃には、1番エントリと連結可能であることを示す、ME₃=“01000000”が格納される。また、どのエントリとも連結できないときは、ME_n=オール“0”になる。125-0〜7は、アドレス一致検出論理15bの出力であり、連結可能な既存のエントリを示し、125-0は0番エントリと連結可能を示し、以下順に、125-7が7番エントリと連結可能を示す。

【0083】図18中の表は、入力A(信号103)、B(信号125-0〜7)による出力Oの値を示す。例えば、AB=“00”のとき、O=0になり、AB=“01”のとき、O=1になる。O=0、1、2に応じて、セクタ501は入力“0”、入力“1”(103)、入力“2”(125-0〜7)を選択する。

【0084】103(0)は、信号線103の0ビット目を表し、以下順に、(7)は7ビット目を表す。信号線103は、セット信号生成論理5b(図17)から出力されるもので、前述した実施例2と同様に(図16)、後段ストアバッファ2cに入力されるストアデータのエントリの番号が出力される。例えば、信号線103の0ビット目が“1”であるときは、入力ストアデータは、データフィールド58bの0番エントリに入力される。

【0085】ME_nの値は、上記したセット信号103、信号線125-0〜7の値により以下のようにセットされる。なお、!=は等しくないことを表す。

(1) 信号線103=オール“0”(つまり、後段ストアバッファに入力がない)

このとき、信号線125-0〜7=オール“0”、すなわち、AB=“00”によって、O=“0”となり、ME_nは過去の値を保持する。

(2) 信号線103!=オール“0”かつ、信号線125-0〜7=オール“0”(つまり、後段ストアバッファに入力があり、連結エントリがない場合)

(a) 信号線103=“1”のエントリは、AB=“10”であるから、O=2となり、セクタ501は、信号線125-0〜7の値を選択し、従って、ME_n=オール“0”となる。

(b) 信号線103=“0”のエントリは、AB=“0

0"であるから、O=0となり、セクタ501は、
"0"を選択し、MENは過去の値を保持する。

(3) 信号線103!=オール"0"、かつ、信号線125-0~7!=オール"0" (つまり、後段ストアバッファに入力があり、連結エントリがある場合)

(a) 信号線103="1"のエントリは、AB="10"であるから、O=2となり、セクタ501は、信号線125-0~7の値を選択し、従って、MENは連結エントリを保持する。例えば、信号線103(0)="1"、つまり0番エントリが"1"であるとき、ME0が選択されて、例えば信号線125-1が"1" (つまり、連結可能なエントリが1番エントリ)であれば、ME0には、ME0="01000000"がセットされる。

(b) 信号線103="0"かつ、信号線125-0~7="0"のエントリは、AB="00"であるから、O=0となり、セクタ501は、"0"を選択し、MENは過去の値を保持する。

(c) 信号線103="0"かつ、信号線125-0~7!="0"のエントリは、AB="01"であるから、O=1となり、セクタ501は、セット信号103-0~7を選択し、MENにセット信号103-0~7を保持する。上記した(3)の(a)の例では、信号線125-1="1"であるので、1番エントリのME1が選択され、ME1にセット信号103(0)、つまり0番エントリがセットされ、ME1="10000000"となる。

【0086】この結果、上記した例では、連結エントリバッファ16には、

ME0="01000000"

ME1="10000000"

がセットされ、0番エントリと1番エントリが連結可能であることが示される。

【0087】図19は、実施例3のアドレス一致検出論理15bの構成を示す図である。データ併合が可能であるか否かはストアアドレスの上位29ビットを比較すればよい。また、データ書き込み幅が本実施例では128ビット(16バイト)であるので、連結可能であるか否かはストアアドレスの上位28ビットを比較すればよい。

【0088】64-0~7は、データ併合が可能であるか否かを判定する29ビットのアドレス比較器、65-0~7は、連結可能であるか否かを判定する28ビットのアドレス比較器である。117はストアアドレス、119-0~7は、アドレスフィールド57bの内容、124-0~7は有効ビット59の内容である。118-0~7は、アドレスが一致しデータ併合が可能なエントリを示す。125-0~7は、アドレスが一致し連結可能なエントリを示す。

【0089】アドレス比較器64-0~7は、enable(124-0~7)が"1" (有効)であるとき、

ストアアドレス117の0~28ビットと、アドレスフィールド57bの内容119-0~7をそれぞれ比較する。アドレスが一致したならば、"1"を出力し、不一致ならば"0"を出力する。enableが"0" (無効)の時は"0"を出力する。

【0090】同様に、アドレス比較器65-0~7は、enable(124-0~7)が"1" (有効)であるとき、ストアアドレス117の0~27ビットと、アドレスフィールド57bの内容119-0~7の0~27ビットをそれぞれ比較する。アドレスが一致したならば、"1"、不一致ならば"1"を出力する。enableが"0" (無効)の時は"0"を出力する。

【0091】図20は、実施例3の後段ストアバッファ2cの構成を示す図である。実施例2の構成にさらに、データセクタ68と、バイトセクタ63bを付加して構成されている。データセクタ68への入力104(28)は、信号線104(SCMストアアドレス)の28ビット目を表す。

【0092】図21は、データセクタ68の構成を示す図である。60k、60lは、データフィールド58bの出力305を選択するセクタであり、60mは連結エントリのセクタであり、67はリードポインタ入れ換え論理である。104(28)はストアアドレス104の28ビット目の値("1"または"0")である。302-0~1は選択されたストアデータ、303-0~1は、データセクタ68内のリードポインタであり、303-0が下位側のリードポインタ、303-1が上位側のリードポインタである。

【0093】同時に2つのエントリを掃き出し可能なように、データセクタは2個設けられている。すなわち、データセクタ60kは下位アドレスのデータを選択し、データセクタ60lは上位アドレスのデータを選択する。

【0094】同時に掃き出すエントリは、掃き出しエントリ306が示すエントリ(これは掃き出し要求111によって決まる)と、そのエントリ306と連結可能なエントリである。セクタ60mは、掃き出しエントリ306の値を基に信号線126-0~7を選択して、当該エントリと連結可能なエントリ番号を出力する。つまり、掃き出しエントリ306の値を持つ連結エントリフィールドMENを参照して、掃き出しエントリ306と連結可能なエントリ番号を出力する。

【0095】本実施例では、SCMのデータ書き込みバスは128ビット(16バイト)幅であり、後段ストアバッファ2cのデータフィールドが64ビット(8バイト)幅であるので、ストアアドレスによってデータをSCMのデータ書き込みバスの上位側アドレスに載せる場合と、下位側アドレスに載せる場合がある。

【0096】掃き出しエントリ306が示すエントリの

ストアアドレスは、図20に示すセクタ60jによって選択され、ストアアドレス104に出力される。このストアアドレス104の28ビット目の信号線104(28)の値が“1”のとき、掃き出しエントリ306のデータを上位側に載せる。そして、連結エントリ(信号線126-0~7)が示すエントリのデータは下位側に載せる。

【0097】逆に、信号線104(28)の値が“0”ならば、掃き出しエントリ306が示すエントリのデータを下位側に、連結エントリが示すエントリのデータを上位側に載せる。

【0098】リードポインタ入れ換え論理67は、上記の条件に基づいてデータセクタ68内のリードポインタ303-0~1を生成する。連結エントリは、前述したように、セクタ60mによって掃き出しエントリ306を基に、連結エントリバッファ16の出力126-0~7を選択する。リードポインタ入れ換え論理67の動作は以下の通りである。

【0099】(1) 信号線104(28) = “1”のとき(入力306、126-0~7の“1”側の入力を選択)

リードポインタ303-0には、セクタ60mの出力(連結エントリ126-0~7)が選択され、セクタ60kに出力され、リードポインタ303-1には、掃き出しエントリ306が選択されてセクタ60lに出力される。従って、掃き出しエントリ306のデータ58bがセクタ60lで選択されて上位側に置かれ、連結エントリのデータ58bがセクタ60kで選択されて下位側に置かれ、データ302-0~1となる。

【0100】(2) 信号線104(28) = “0”のとき(入力306、126-0~7の“0”側の入力を選択)

リードポインタ303-0には、掃き出しエントリ306が選択されてセクタ60kに出力され、リードポインタ303-1には、セクタ60mの出力(連結エントリ)が選択されてセクタ60lに出力される。従って、掃き出しエントリ306のデータが下位側に置かれ、連結エントリのデータが上位側に置かれて、データ302-0~1となる。

【0101】図22は、バイトセクタ63bの構成を示す図である。60r、60qは、リードポインタ303-0~1を基に、後段バイトマスクバッファの内容116-0~7から掃き出すエントリのバイトマスク304-0~1を取り出すセクタである。60n、60pは、バイトマスク304-0~1の値を基に有効なバイト位置のデータを取り出すセクタである。

【0102】例えば、掃き出しエントリ306のデータ58bが上位側にあるときは、リードポインタ303-1によってセクタ60rが働き、掃き出しエントリ306に対応するバイトマスクを選択し、セクタ60p

に出力する。セクタ60pは、掃き出しエントリ306のデータ58bの内、バイトマスク304-1で指示された有効なバイト位置のデータを取り出し、SCMストアデータ105として出力する。

【0103】〈実施例4〉次に、実施例4について説明する。キャッシュミスにより処理が止まることを回避する目的で、キャッシュメモリからではなく直接主記憶からデータを読み込み、ロードデータの到着を待たずに次の処理に移ることができるロード命令を導入する。本実施例では、このロード命令をpreload命令と呼ぶことにする。preload命令は大量のデータを扱う場合に有利である。

【0104】しかし、同じアドレスに対するストア命令とpreload命令があった場合、前のストア命令をpreload命令が追い越してしまうと、preload命令は間違ったデータを読み込んでしまう。そこで、主記憶を参照する命令間の順序性を保障する必要がある。

【0105】図23は、本発明の実施例4の構成を示す図である。17は本実施例によって設けられたロードストアバッファである。ロードストアバッファ17は、preload命令とストア命令を保持するFIFO型のバッファである。

【0106】図24は、ロードストアバッファの構成を示す図である。66は命令フィールドOPである。命令フィールド66は、当該エントリの命令がpreload命令かストア命令かを示す。

【0107】ロードストアバッファ17は、本実施例では、8段のエントリからなり、各エントリは32ビットのアドレスフィールド、64ビットのデータフィールド、命令フィールドOPからなる。ストアアドレスとデータは信号線108から入力され、命令は信号線127から入力され、ライトエントリ122の示すエントリに書き込む。このとき、preload命令ならば、データフィールドの値は意味を持たない。一方、ロードストアバッファ17からの掃き出しは、リードエントリ123の示すエントリの、アドレス、データ、命令をセクタ60eで選択する。

【0108】選択した命令がpreload命令ならば、主記憶に対して該命令が出力されるだけで、パイプライン遅延論理3には出力されない。ストア命令ならば、前述した実施例と同様に、主記憶に書き込むと同時に、パイプライン遅延論理3に送りSCMヒット判定を行う。

【0109】

【発明の効果】以上、説明したように、本発明によれば、ストアバッファの入力と出力のピッチを一致させることができるので、ストアバッファに空きがなくなる状態の発生を低減することができ、その結果、CPUの性能低下を防ぐ効果がある。また、キャッシュヒットする

ストア命令のみストアバッファに入力するので、ストアバッファを効率よく使用できる効果がある。さらに、複数のストア命令によるデータ書き込みを同時に処理できるので、スループットを向上する効果がある。また、ストアスルー方式のキャッシュメモリを採用した場合、ストアバッファへの入力と主記憶への書き込みピッチが一致するので、主記憶に繋がるバスを効率よく使用できる効果がある。

【図面の簡単な説明】

【図1】本発明の実施例1の構成を示す図である。

【図2】実施例1の前段ストアバッファの構成を示す図である。

【図3】SCMヒット判定論理の動作例を示す図である。

【図4】実施例1のセット信号生成論理の構成を示す図である。

【図5】リードライトポインタの構成を示す図である。

【図6】実施例1の後段ストアバッファの構成を示す図である。

【図7】実施例1の動作例を示す図である。

【図8】本発明の実施例2の構成を示す図である。

【図9】実施例2の前段ストアバッファの構成を示す図である。

【図10】実施例2の前段バイトマスクバッファの構成を示す図である。

【図11】バイトマスク更新論理の構成を示す図である。

【図12】後段バイトマスクバッファの構成を示す図である。

【図13】実施例2の後段ストアバッファの構成を示す図である。

【図14】バイトセクタの構成を示す図である。

【図15】アドレス一致論理の構成を示す図である。

【図16】実施例2のセット信号生成論理の構成を示す図である。

【図17】本発明の実施例3の構成を示す図である。

【図18】連結エントリバッファの構成を示す図である。

【図19】実施例3のアドレス一致論理の構成を示す図である。

【図20】実施例3の後段ストアバッファの構成を示す図である。

【図21】データセクタの構成を示す図である。

【図22】バイトセクタの構成を示す図である。

【図23】本発明の実施例4の構成を示す図である。

【図24】ロードストアバッファの構成を示す図である。

【図25】本発明の実施対象となる情報処理装置の構成図である。

【図26】ダイレクトマッピング法を説明する図である。

る。

【符号の説明】

1 a～b 前段ストアバッファ

2 a～c 後段ストアバッファ

3 SCMヒット判定論理

4 主記憶

5 a、b セット信号生成論理

6 データアレイ

7 アドレスアレイ

8 バイブライン遅延論理

9 アドレス比較器

11 前段バイトマスクバッファ

12 後段バイトマスクバッファ

13 バイトマスク遅延論理

14 バイトマスク更新論理

15 a、b アドレス一致検出論理

16 連結エントリバッファ

17 ロードストアバッファ

51 後段ストアバッファライトポインタ

52 更新論理

53 アドレスフィールド

54 a、b データフィールド

55 リードライトポインタ

56 a ライトポインタ用ラッチ

56 b リードポインタ用ラッチ

57 a、b 後段ストアバッファアドレスフィールド

58 a、b 後段ストアバッファデータフィールド

59 有効ビット

60 a～r セクタ

61 後段ストアバッファリードポインタ

62 後段ストアバッファリードポインタ用ラッチ

63 a、b バイトセクタ

64、65 アドレス比較器

66 命令フィールド

67 リードポインタ入れ換え論理

68 データセクタ

69 デコーダ

71 命令実行部

72 1次キャッシュメモリ (FCM)

73 スタアバッファ

74 CPU

75 2次キャッシュメモリ (SCM)

91 a～f ラッチ

101 SCMアドレスアレイ参照アドレス

102 SCMアドレスアレイの内容

103 セット信号

104 SCMストアアドレス

105 SCMストアデータ

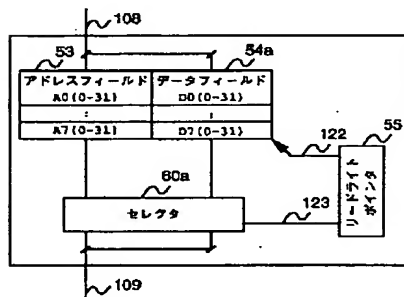
106 主記憶書き込みアドレスとデータ

107 SCMヒット判定結果

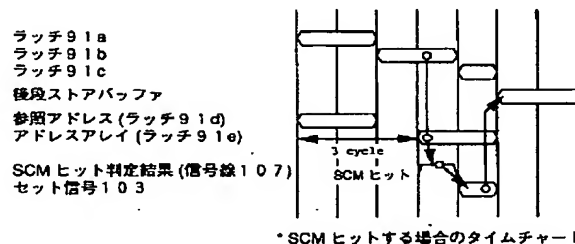
108 ストアアドレスとデータ
 109 前段ストアバッファから出力したストアアドレスとデータ
 110 遅延したストアアドレスとデータ
 111 掃き出し要求
 112 前段バイトマスク入力
 113 前段バイトマスク出力
 114 バイトマスク遅延論理出力
 115-0~7 後段バイトマスク入力
 116-0~7 後段バイトマスク出力
 117 アドレス一致入力
 118-0~7 アドレス一致出力
 119-0~7 アドレスフィールドの内容
 121 データ併合の有無

122 ライトエントリ
 123 リードエントリ
 124-0~7 有効ビット
 125-0~7 連結エントリ
 126-0~7 連結エントリバッファ出力
 127 命令
 128 ライト要求
 129 リード要求
 301、302 セレクトしたデータ
 303 セレクト信号
 305 データフィールド
 306 掃き出しエントリ
 307 データフィールドセット信号
 308 選択したバイトマスク

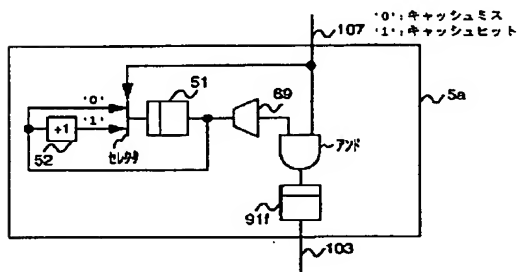
【図2】



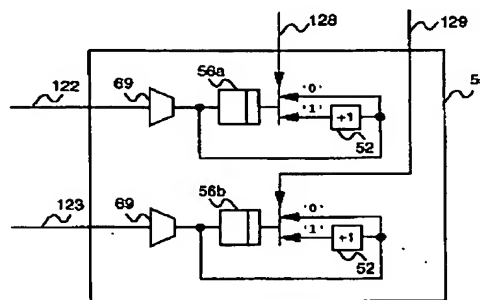
【図3】



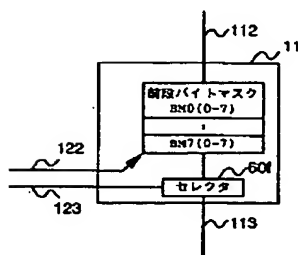
【図4】



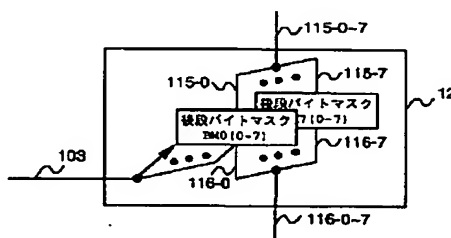
【図5】



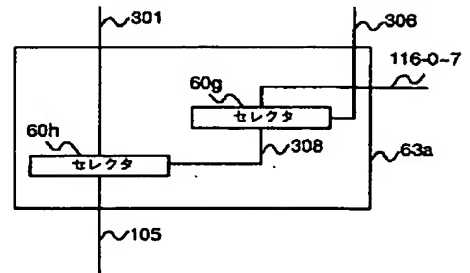
【図10】



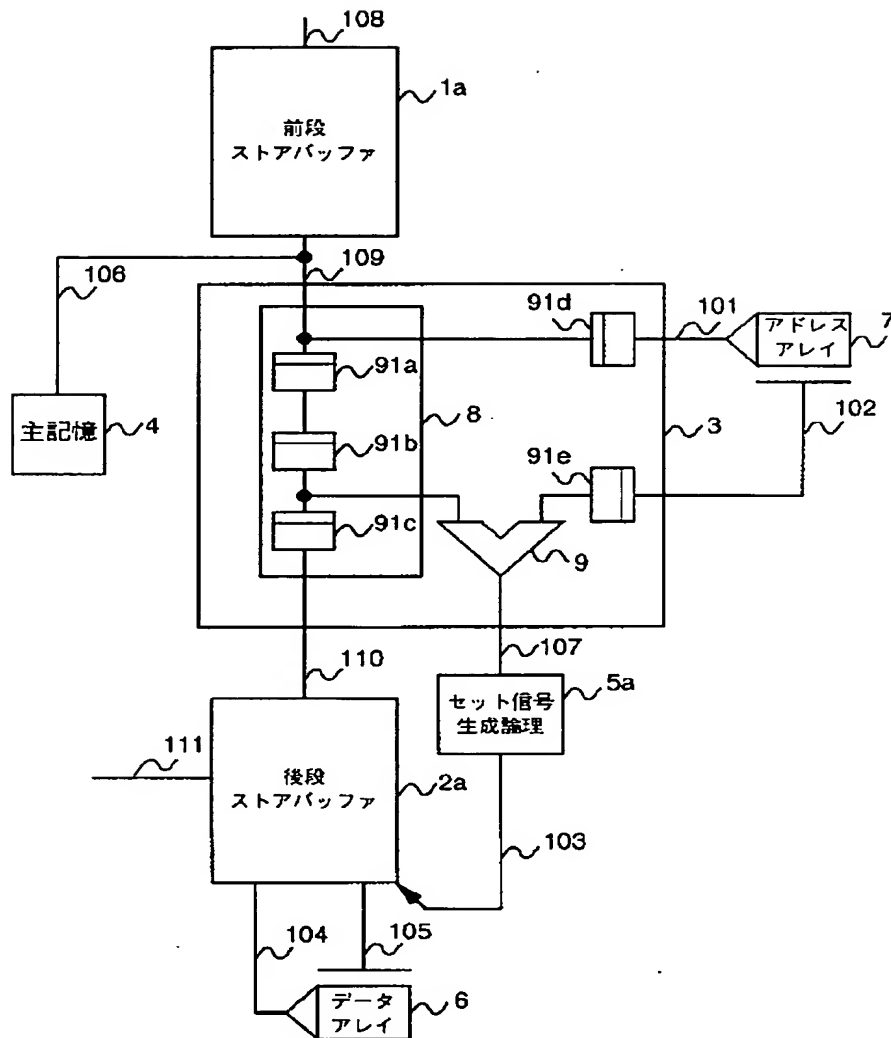
【図12】



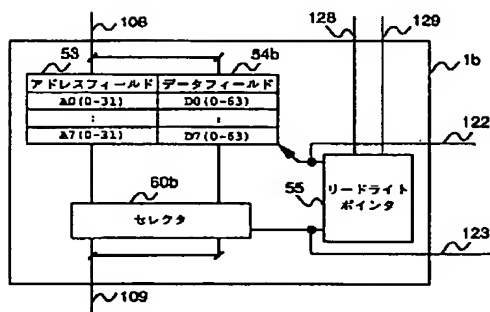
【図14】



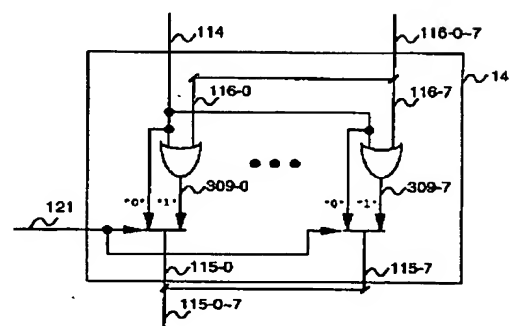
【図1】



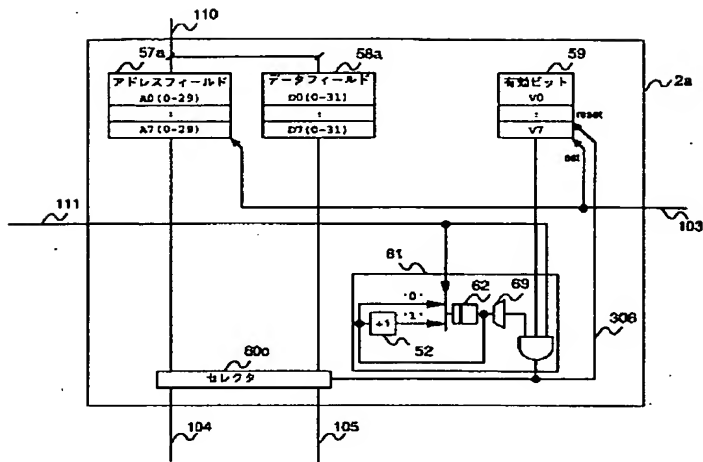
【図9】



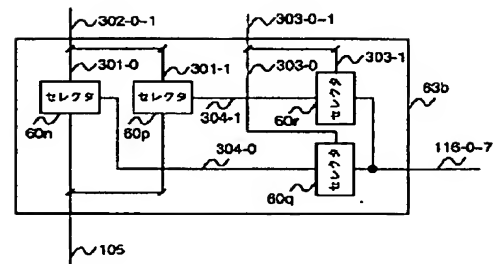
【図11】



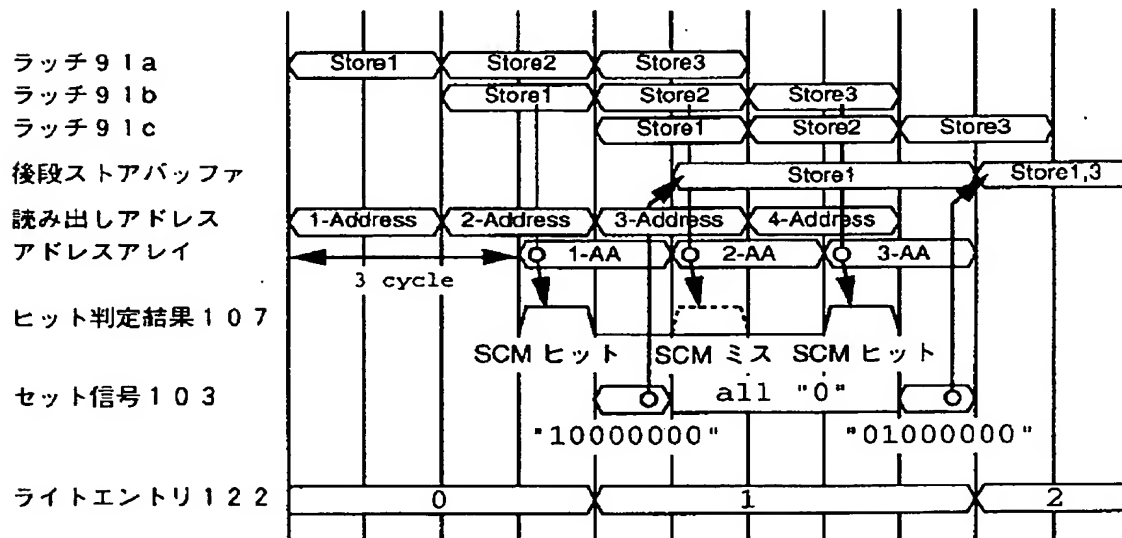
【図6】



【図22】

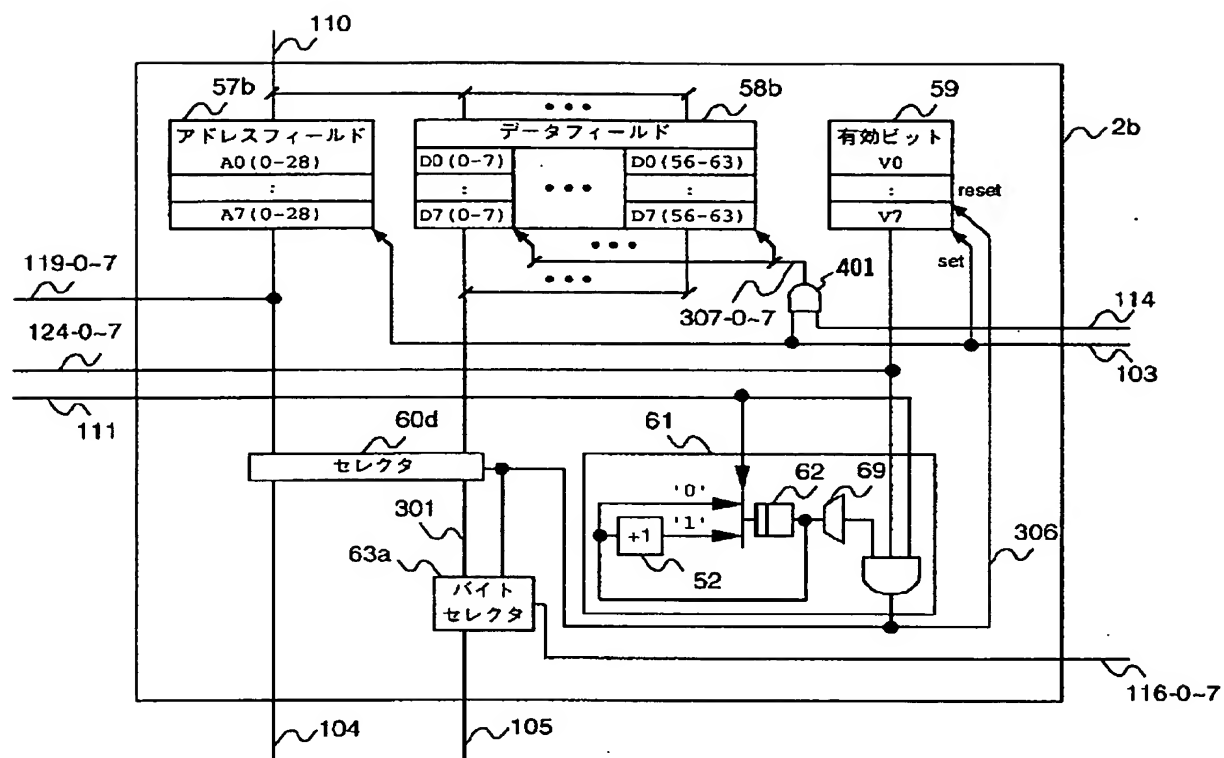


【図7】

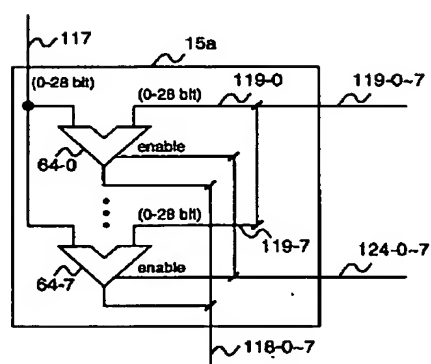


Store n : n 番目のストア命令のストアアドレスとデータ
 n-Address : n 番目のストア命令のアドレス
 n-AA : n 番目のストア命令のアドレスアレイ

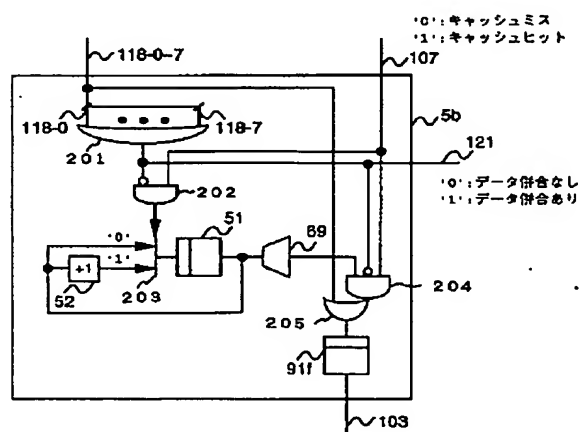
【図13】



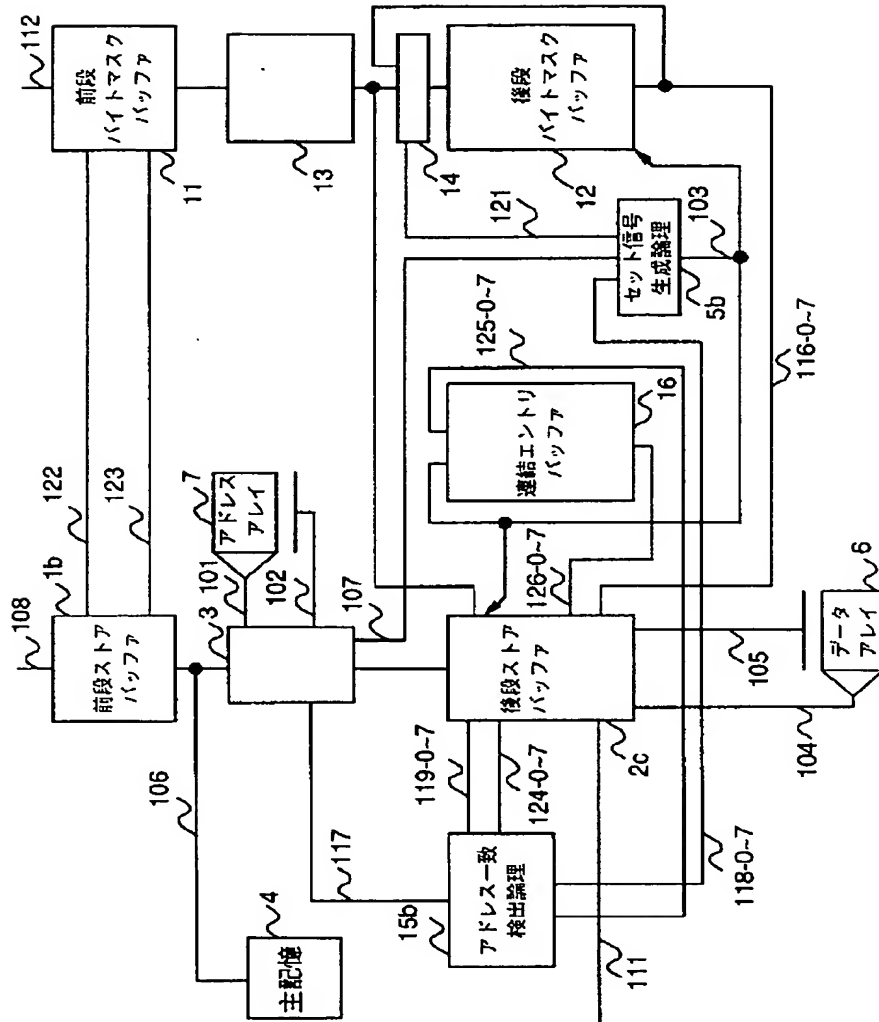
【図15】



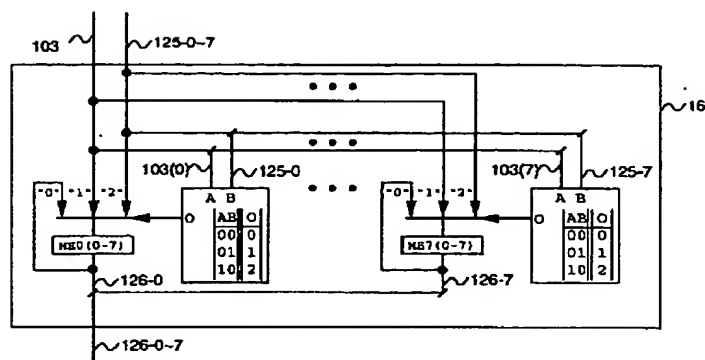
【図16】



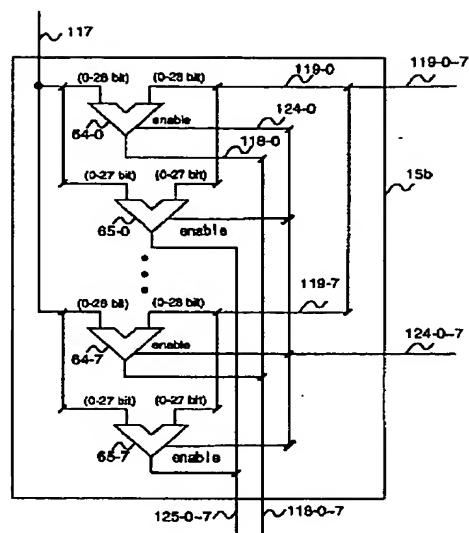
【図17】



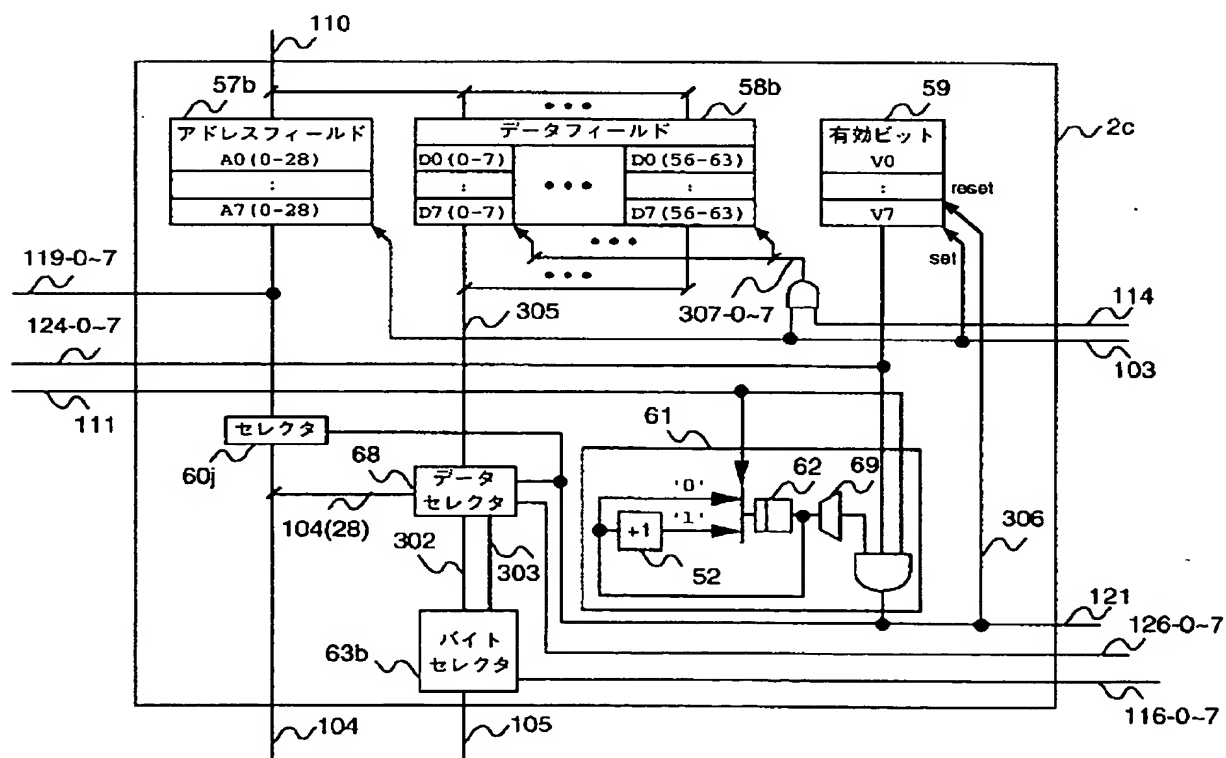
【図18】



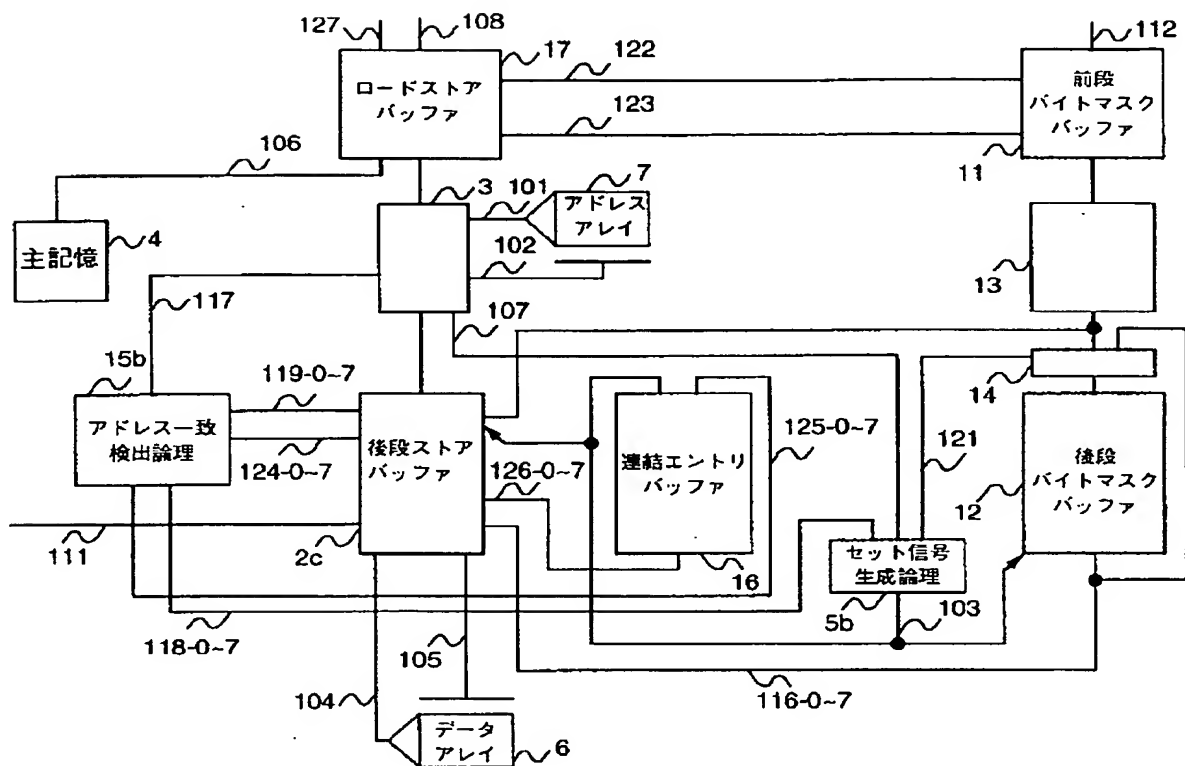
【図19】



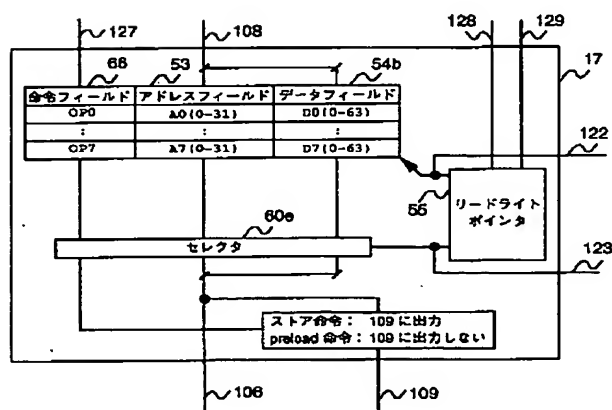
【図20】



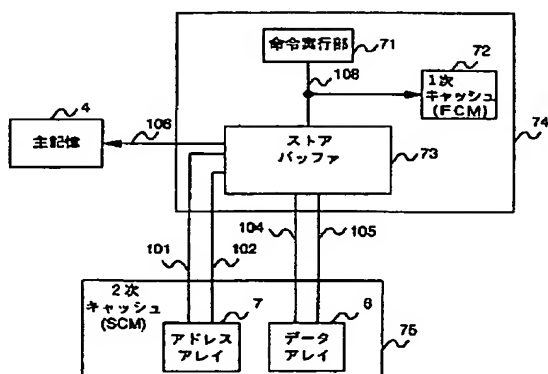
【图23】



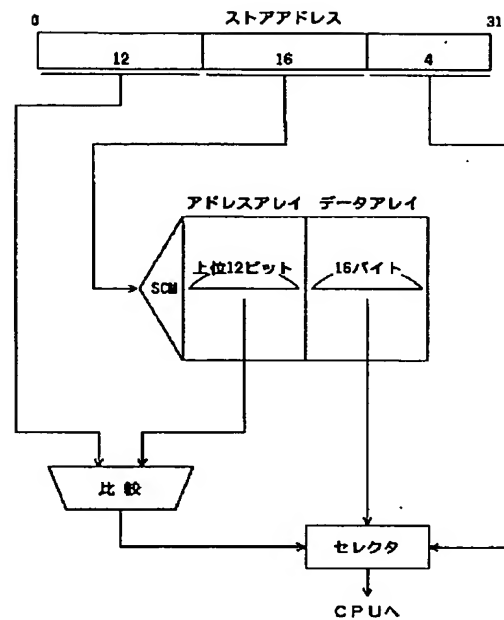
【图24】



【图25】



【図26】



フロントページの続き

(72)発明者 釜田 栄樹
神奈川県秦野市堀山下1番地 株式会社日立製作所汎用コンピュータ事業部内

(72)発明者 磯部 敏子
神奈川県秦野市堀山下1番地 日立コンピュータエンジニアリング株式会社内

(72)発明者 山本 敬
神奈川県秦野市堀山下1番地 株式会社日立製作所汎用コンピュータ事業部内

(72)発明者 上原 克利
神奈川県秦野市堀山下1番地 株式会社日立製作所汎用コンピュータ事業部内